

Class 6: R functions

Junlin Ruan (PID: A17839687)

All functions in R have at least 3 things:

- A **name**, we pick this and use it to call the function.
- Input **arguments**, there can be multiple comma separated inputs to the function.
- The **body**, lines of R code that do the work of the function.

Our first wee function:

```
add <- function(x, y=1) { #default if set to = sth
  x + y
}
```

Let's test our function

```
add(c(1,2,3), y=10)
```

```
[1] 11 12 13
```

```
add(10)
```

```
[1] 11
```

```
add(10,100)
```

```
[1] 110
```

A second function

Let's try something more interesting. Make a sequence generation tool.

The `sample()` function could be useful here.

```
sample(1:10, size = 3)
```

```
[1] 5 1 4
```

change this to work with the nucleotide A C G and T and return 3 of them

```
n <- c("A", "C", "G", "T")
sample(n, size = 15, replace = TRUE)
```

```
[1] "T" "T" "T" "A" "T" "C" "A" "A" "G" "G" "C" "C" "C" "C" "G"
```

Turn this snippet into a function that returns a user specified length DNA sequence. Let's call it `generate_dna()`...

```
generate_dna <- function(len=10, fasta=FALSE) {
  n <- c("A", "C", "G", "T")
  v <- sample(n, size = len, replace = TRUE)

  # Make a single element vector
  s <- paste(v, collapse="")
  #stick output together with "" between
  cat("Well done you!\n") #\n = new line

  if(fasta){
    return( s )
  } else {
    return( v )
  }
}
```

```
generate_dna(5)
```

```
Well done you!
```

```
[1] "A" "A" "G" "G" "C"
```

```
s <- generate_dna(15)
```

Well done you!

```
s
```

```
[1] "G" "C" "C" "A" "G" "A" "A" "T" "T" "C" "C" "G" "G" "G" "C"
```

I want the option to return a single element character vector with my sequence all together like this: “GGAGTAC”

```
generate_dna(10, fasta=TRUE)
```

Well done you!

```
[1] "ACTGGGCTAA"
```

```
generate_dna(10, fasta=FALSE)
```

Well done you!

```
[1] "C" "G" "C" "A" "T" "G" "G" "A" "T" "T"
```

A more advanced example

Make a third function that generates protein sequence of a user specified length and format.

```
generate_protein <- function(size=15, fasta=TRUE) {  
  aa <- c("A", "R", "N", "D", "C", "E", "Q", "G",  
         "H", "I", "L", "K", "M", "F", "P", "S",  
         "T", "W", "Y", "V")  
  
  seq <- sample(aa, size = size, replace = TRUE)  
  
  if(fasta){  
    return(paste(seq, collapse = ""))  
  }  
}
```

```
    } else {
      return(seq)
    }
}
```

Try this out...

```
generate_protein(10)
```

```
[1] "AELWIMFWCQ"
```

Q. Generate random protein sequences between lengths 5 and 12 amino acids

```
generate_protein(5)
```

```
[1] "SYCVE"
```

```
generate_protein(6)
```

```
[1] "PRGTIW"
```

One approach is to do this by brute force calling our function for each length 5 to 12.

Another approach is to write a `for()` loop to iterate over the input values 5 to 12.

A very useful third R specific approach is to use the `sapply()` function.

```
seq_lengths <- 5:12
for (i in seq_lengths) {
  cat(">", i, "\n", sep="")
  cat(generate_protein(i))
  cat("\n")
}
```

```
>5
WIGSL
>6
YDSTVV
>7
PRFERVC
```

```
>8  
RWETMKPD  
>9  
DCTRACLT  
>10  
DQHNQFSNKD  
>11  
CFTLQMWEQHW  
>12  
TLGSNLHAKQTS
```

```
sapply(5:12, generate_protein)
```

```
[1] "IFLSA"          "AFQCAD"        "SCTMQHI"       "QEATCMQP"      "TLDMWIMAE"  
[6] "RDCFIIIGSR"   "NRRKPCECHDY"  "LQWDDVTNNQPA"
```

Key-point: Writing functions in R is doable but not the easiest thing. Starting with a working snippet of code and then using LLM tools to improve and generalize your function code is a productive approach.